(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 944 173 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
22.09.1999 Bulletin 1999/38

(51) Int Cl.[6]: **H03M 13/00**

(21) Application number: **99302042.9**

(22) Date of filing: **17.03.1999**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **17.03.1998 GB 9805573**

(71) Applicant: **SAMSUNG ELECTRONICS CO., LTD.**
Suwon-City, Kyungki-do (KR)

(72) Inventor: **Lee, Sang-Bong**
Songnam-shi, Kyonggi-Do (KR)

(74) Representative:
Tunstall, Christopher Stephen et al
Dibb Lupton Alsop,
Fountain Precinct
Balm Green, Sheffield S1 1RZ (GB)

(54) Add-compare selection circuit for a Viterbi decoder

(57) A high-speed add-compare selection apparatus, for a Viterbi algorithm processing apparatus having a branch metric or Euclidean calculator and a metric memory, is described. First and second previous metric values are supplied from the metric memory to first and second registers. The first previous metric value from the first register and a branch metric or Euclidean value of the present state calculated by the branch metric or Euclidean calculator are added, as are the second previous metric value from the second register and a branch metric value of the next state calculated by the branch metric calculator. The values obtained are compared and a survival metric value calculated accordingly.

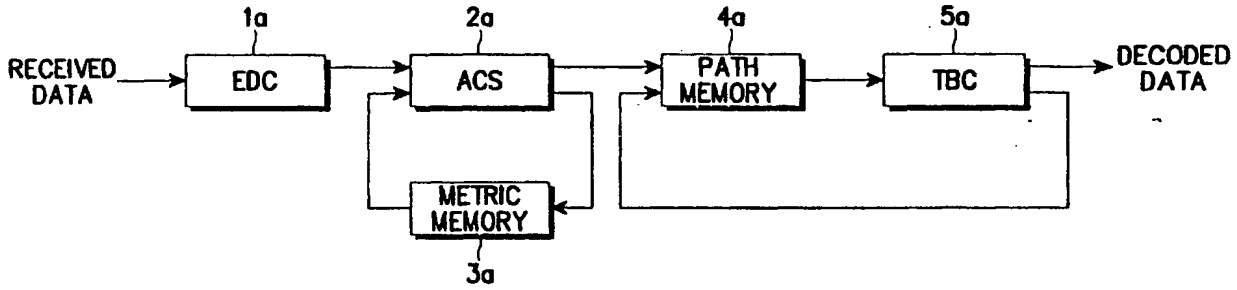

FIG. 3A

EP 0 944 173 A2

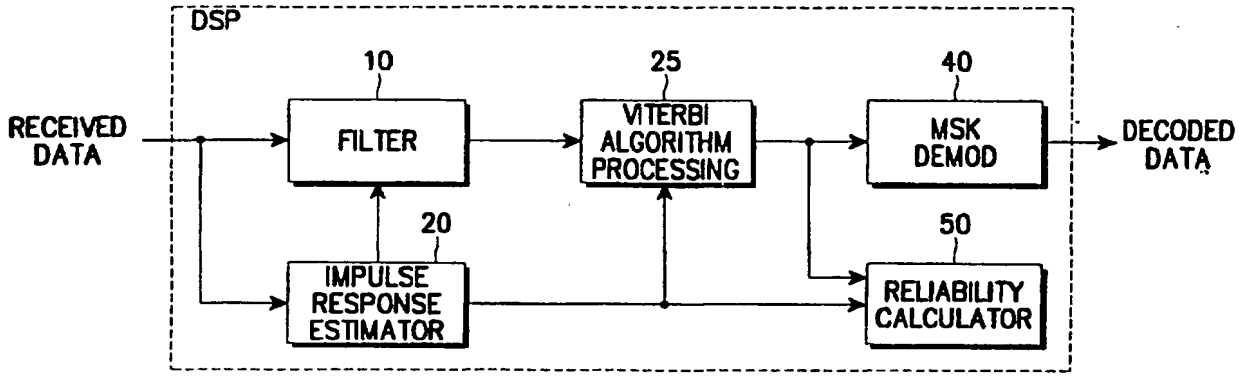
DSP
10
25
40
RECEIVED DATA
FILTER
VITERBI ALGORITHM PROCESSING
MSK DEMOD
DECODED DATA
20
50
IMPULSE RESPONSE ESTIMATOR
RELIABILITY CALCULATOR

FIG. 3B

Description

BACKGROUND OF THE INVENTION

[0001] The present invention relates to an apparatus and method for processing a Viterbi algorithm for example in a digital mobile communications system.

[0002] Viterbi algorithms can easily be used to implement a sequence determining method modelling a Gaussian channel. Conventionally, Viterbi algorithms are widely used owing to their excellent error correction rates. In Viterbi decoding, the distance of a received set of data to the data of all other possible paths on a trellis (each node of the trellis in a cycle "n" representing a state of the received set of data in that cycle) is determined and the possible path closest to the received path is selected. Once a surviving path has been decided upon, the corresponding sequence of bits, now decoded, can be traced back along this path. However, it does involve a lot of calculations and implementation time. In particular, the add-compare select (ACS) portion and trace-back portion require the greatest amount of time. If a constraint length "K" is applied in a convolutional encoder, the number of states, calculated by the ACS portion, is determined. Also, the path length of the trace-back portion, determined by simulation, plays an important role in determining the performance of a Viterbi algorithm.

[0003] In various mobile communication terminals which use TDMA, such as GSM, which is a European digital mobile communications standard, since the time for processing received data is predetermined, the Viterbi algorithm must be as fast as possible. For example, since a TDMA cycle of the GSM system is limited to 4.615 ms, it is most important to secure a time margin for the purpose of achieving stable operation.

[0004] Recently, baseband systems have often been implemented by a DSP. However, since the Viterbi algorithm processing portion requires a lot of calculations and fast processing speed, a separate co-processing system is necessary. Although the Viterbi algorithm is determined in advance as described above, there is still the possibility of improvements in design which increase speed and efficiency.

[0005] FIG. 1 is a block diagram of a conventional Viterbi decoder. A branch metric calculator (BMC) 1 receives a digital signal and calculates a branch metric value as probabilistic information. An add-compare selector (ACS) 2b inputs the branch metric value from the BMC 1 and updates a previous path metric corresponding to each state in a trellis, using the input branch metric value. The ACS 2b compares the updated path metrics with each other to determine the most likely one and outputs a selected path metric and a determining bit. In a metric memory 3b, the path metric selected by the ACS 2b is fed back to the ACS 2b in a subsequent step. A path memory 4b stores the determining bit output from the ACS 2b. A trace-back controller 5b implements a trace-back operation using the determining bit stored in the path memory 4b and traces back the sequence of original information.

[0006] The conventional design of the ACS 2b will now be described. For the sake of explanation, an example of 4 states will be used. It is necessary to trace the most likely metric in the Viterbi algorithm for searching proper data ie for determining the most likely correct data from the received data. The following formula is used for calculating a survival metric in the Viterbi algorithm over four states (00,01,10,11):

$$M_n,s0 = \max(M_{n-1}{}^{p0} + bmc1^{p0}{}_{s0} \bullet M_{n-1}{}^{p1} + bmc2^{p1}{}_{s0})$$

$$M_n,s1 = \max(M_{n-1}{}^{p2} + bmc1^{p2}{}_{s1} \bullet M_{n-1}{}^{p3} + bmc2^{p3}{}_{s1})$$

$$M_n,s2 = \max(M_{n-1}{}^{p0} + bmc1^{p0}{}_{s2} \bullet M_{n-1}{}^{p1} + bmc2^{p1}{}_{s2})$$

$$M_n,s3 = \max(M_{n-1}{}^{p2} + bmc1^{p2}{}_{s3} \bullet M_{n-1}{}^{p3} + bmc2^{p3}{}_{s3})$$

where M implies the survival metric of two metric values, $M_{n-1}{}^{p0}$ is an (n-1)th cycle survival metric for the 00 state, $M_{n-1}{}^{p1}$ is an (n-1)th cycle survival metric for the 01 state and so on; Mns0 is the selected survival metric for the nth cycle in the 00 state, Mnsl is the selected survival metric for the nth cycle in the 11 state, etc.; and $bmc1^{p0}{}_{s0}$ represents a first branch metric value (transition probability) to a state s0 (ie 00 in this case) from a state p0 (also 00 in this case) and $bmc2^{p1}{}_{s0}$ represents a second branch metric value to a state s0 but from a p1 (ie 11 state in this case).

[0007] The BMC 1b generates the same metric value as that of the convolutional encoder and obtains the difference from received data. Thus, the survival metric of the present state is the larger value of two values, i.e., a previous metric value and a value obtained by adding the previous metric value and the survival metric of the present state. To this end, at least two adders and a comparator are necessary.

[0008] To calculate the metric value of the present state, the previous metric value stored in the metric memory 3b is read and compared with a value obtained by adding the read metric value and the present metric value. Then the most similar value to the transmitted value is searched.

[0009] FIG. 2 illustrates metric values of the respective states and calculated branch metric values for calculating the survival metric of the Viterbi algorithm. As shown, to calculate the metric value of a state 00, the previous metric value $M_{n-1}{}^{p0}$ is read. Then, a value obtained by adding the same to $bmc1^{p0}{}_{s0}$ is compared with a value obtained by adding $M_{n-1}{}^{p1}$ to $bmc2^{p1}{}_{s0}$. Of the two values, the larger value is determined as the survival metric (M_n,s0) of the state 00. Thus, to obtain one survival metric value, 2 cycles are required in each state just for reading the previous metric values. In other words, to calculate the survival metric value, the previous metric values 0 and 1 must be read in the state 00, and the previous metric values 2 and 3 must be read in the state 01. When repeating such operation, 8 cycles are required only for reading the previous metric values to calculate the survival metric values in the case of four states.

[0010] However, according to this method, since the memory must frequently be accessed, high power consumption occurs. Also, since many clocks are used, this method is not suitable for high speed Viterbi algorithm implementation, such as GSM.

[0011] In the meantime, such a Viterbi algorithm processing apparatus is used for both a Viterbi equalizer and a Viterbi decoder. FIG. 3A illustrates a block diagram of a Viterbi algorithm processing apparatus included in the Viterbi equalizer, and FIG. 3B illustrates a block diagram of the Viterbi equalizer including the Viterbi algorithm processing apparatus.

[0012] Referring to FIG. 3B, an impulse response estimator 20 receives input data and measures a channel impulse response of the received data. A filter 10 is implemented by a finite impulse response (FIR) Filter designed to have the maximum signal-to-noise ratio at the output terminal at a particular time. The filter 10 is a matched filter which multiplies a reversal of the channel impulse response input from the impulse response estimator 20 by the received data and then time-shifts the multiplied value. A Viterbi algorithm processing apparatus 25 receives the data output from the filter 10 and the channel impulse response from the impulse response estimator 10 and performs the Viterbi algorithm for equalization. A demodulator 40 MSK demodulates data output from the Viterbi algorithm processing apparatus 25. A reliability calculator 50 calculates a reliability of the data processed in the Viterbi algorithm processing apparatus 25.

[0013] Such a Viterbi equalizer can be implemented by a DSP (Digital Signal Processor) or by hardware (eg, an equalizer processor or a VLSI). However, when realized by the DSP, the Viterbi equalizer has the increased calculations for Euclidean distance of the Viterbi algorithm, and the increased bit operations at the ACS and trace-back portions, causing numerous manipulations and increasing the power consumption. In addition, to satisfy the system timing, some DSP makers provide coprocessors. When realized by hardware, multipliers, dividers and adders are required according to the characteristics of the filter or the Viterbi algorithm, raising a complexity problem. Also, when the Viterbi equalizer is implemented by an ASIC (Application Specific Integrated Circuit), there may be raised a chip size problem.

Summary of the Invention

[0014] An objective of the present invention is to provide an apparatus and method for processing a Viterbi algorithm in a Viterbi decoder, which increases the processing speed of an ACS portion requiring a large quantity of calculation, thus reducing power consumption.

[0015] Another objective of the present invention is to provide a small high-speed Viterbi algorithm processing apparatus by divisionally designing the Viterbi algorithm into a portion to be processed by a digital signal processor and another portion to be processed by hardware according to the contents of the algorithm so as to increase the efficiency.

[0016] Accordingly the invention provides a high-speed add-compare selection apparatus, for a Viterbi algorithm processing apparatus having a branch metric calculator or a Euclidean value calculator or the like and a metric memory, comprising:

first and second registers;
means for supplying to the first and second registers first and second previous metric values from the metric memory;
a first adder for adding the first previous metric value from the first register and a branch metric value or a Euclidean value or the like of the present state calculated by the calculator;
a second adder for adding the second previous metric value from the second register and a branch metric value or a Euclidean value or the like of the next state calculated by the calculator; and
a comparator for comparing the outputs of the first and second adders and calculating a survival metric value accordingly.

[0017] Preferably, the means for supplying the first and second previous metric values to the first and second registers

comprises means for supplying to the first register a first previous metric value from the second register and for supplying to the second register a second previous metric value.

[0018] Preferably, the first and second metric values are Hamming distances.

[0019] Preferably, the first and second metric values are Euclidean distances.

[0020] Preferably, the first adder is for adding the first previous metric value input from the first register to a Euclidean value of a next state calculated by the Euclidean value calculator and the second adder is for adding the second previous metric value input from the second register to the Euclidean value of the next state calculated by the Euclidean value calculator.

[0021] In a still further aspect of the invention there is provided a Viterbi algorithm processing method for use in a processing apparatus having a branch metric calculator or a Euclidean value calculator or the like and a metric memory, comprising:

supplying to first and second registers first and second previous metric values from the metric memory;
calculating a first survival metric value of a present state using the first and second previous metric values from the first and second registers, a branch metric value or Euclidean value or the like of the present state calculated by the calculator and a branch metric value or Euclidean value or the like of the next state calculated by the calculator;
calculating a second survival metric value of another present state using the first and second previous metric values from the first and second registers, a branch metric value or a Euclidean value or the like of the present state calculated by the calculator and a branch metric value or a Euclidean value or the like of the next state calculated by the calculator;
supplying to the first and second registers third and fourth previous metric values from the metric memory; and
calculating a third survival metric value of another present state using the third and fourth previous metric values from the first and second registers, a branch metric value or Euclidean value or the like of the present state calculated by the calculator and a branch metric value or a Euclidean value or the like of the next state calculated by the calculator.

[0022] In a further aspect of the invention there is provided a Viterbi algorithm processing method comprising:
calculating a fourth survival metric value of another present state using the third and fourth previous metric values from the first and second registers, a branch metric value or Euclidean value or the like of the present state calculated by the branch metric calculator and a branch metric value or Euclidean value or the like of the next state calculated by the calculator.

[0023] The survival metric values of the present states may be calculated by:

adding the previous metric value from the first register and a branch metric value or Euclidean value or the like of the present state calculated by the calculator to provide a first added value;
adding the previous metric value from the second register and a branch metric value or a Euclidean value or the like of the next state calculated by the calculator to provide a second added value; and
comparing the first and second added values and calculating a survival metric value accordingly.

[0024] In a further aspect of the invention there is provided a Viterbi algorithm processing method for use in a processing apparatus having a branch metric or Euclidean value calculator or the like, a metric memory, first and second registers and an add-compare selector for comparing a value obtained by adding a first previous metric value to a branch metric or Euclidean value or the like of a present state calculated by the calculator with a value obtained by adding a second metric value to a branch metric, Euclidean value or the like of a next state calculated by the calculator to determine a higher value of them as a survival metric, comprising the steps of:

(a) reading the first and second previous metric values from the metric memory and storing the read values in the first and second registers to calculate a metric value of the present state;

(b) calculating a metric value of another present state using the first and second previous metric values of the first and second registers; and, preferably,

(c) returning to said step (a) to calculate a metric value of another present state using a previous metric value being different from the first and second metric values of the first and second registers.

[0025] In a further aspect of the invention there is provided a Viterbi algorithm processing method for use in an add-compare selector having first and second registers, a branch metric or Euclidean value calculator or the like and a

metric memory, comprising the steps of:

(a) reading n-th (where n = 1, 2, 3,....) and (n+1)-th previous metric values from the metric memory and storing the read values in the first and second registers;

(b) comparing a value obtained by adding the n-th previous metric value stored in the first register to a branch metric value, Euclidean value or the like of the present state calculated by the calculator with a value obtained by adding the (n+1)-th previous metric value stored in the second register to a branch metric, Euclidean value or the like of a next state calculated by the calculator so as to determine a higher value of them as an n-th survival metric;

(c) comparing a value obtained by adding the n-th previous metric value stored in the second register to another branch metric or Euclidean value or the like of the present state calculated by the calculator with a value obtained by adding the (n+1)-th previous metric value stored in the second register to another branch metric value or Euclidean value or the like of the next state calculated by the calculator so as to determine a higher value of them as an (n+1)-th survival metric; and

(d) increasing n by one and returning to said step (a) to process the Viterbi algorithm for another present state.

[0026] In a further aspect of the invention there is provided a Viterbi equalizer comprising:

a Viterbi pre-processing portion for estimating a channel impulse response from received data and for filtering the data using the impulse response using a digital signal processor; and

a Viterbi algorithm processing apparatus for performing a Viterbi algorithm for equalization of the filtered data and the estimated channel impulse response provided from the Viterbi pre-processing portion.

[0027] Preferably, the pre-processing portion multiplies a reversal of the estimated channel impedance response by the received data, and time-shifts the multiplied result.

[0028] Preferably, a Viterbi equalizer comprises a high speed add-compare selection apparatus as described herein.

Brief Description of the Drawings

[0029] The present invention will now be described by way of example with reference to the accompanying drawings in which:

[0030] FIG. 1 is a block diagram of a conventional Viterbi decoder.

[0031] FIG. 2 illustrates metric values of each state and calculated branch metric values for calculating the survival metric of a Viterbi algorithm.

[0032] FIG. 3A is a block diagram of a Viterbi algorithm processing apparatus included in the Viterbi equalizer.

[0033] FIG. 3B is a block diagram of the Viterbi equalizer including the Viterbi algorithm processing apparatus of FIG. 3A.

[0034] FIG. 4 is a schematic diagram of an add-compare selector of a Viterbi decoder according to a preferred embodiment of the present invention.

[0035] FIG. 5 is an operational timing diagram according to a preferred embodiment of the present invention.

[0036] FIG. 6 is a diagram for explaining how to implement the Viterbi equalizer according to an embodiment of the present invention.

Detailed Description of the Preferred Embodiment

[0037] FIG. 4 is a schematic diagram of an add-compare selector of a Viterbi decoder according to a preferred embodiment of the present invention. In FIG. 4, by replacing bmc1 and bmc2 with edc1 and edc2, it is possible to implement the ACS of a Viterbi equalizer.

[0038] The add-compare selector of FIG. 4 includes a register portion 10 comprising a second register 10B for storing a predetermined previous metric value read from the metric memory 3b shown in FIG. 1 and a first register 10A for storing the previous metric value shifted and input from the second register 10B. A first adder 30 adds the first previous metric value input from the first register 10A to a calculated branch metric value bmcl of the present state, output from the BMC shown in FIG. 1. A second adder 35 adds the second previous metric value input from the second register 10B to a calculated branch metric value bmc2 of the next state, also output from the BMC shown in FIG. 1. A comparator 40 compares the outputs of the first and second adders 30 and 35 to determine the larger value as the survival metric,

i.e., the present metric value PM2.

[0039] The first and second branch metric values bmcl and bmc2 may be Euclidian distances in an equalizer or Hamming distances in a convolutional decoder. The BMC 1b shown in FIG. 1, used in the present invention, obtains the difference between received data and predetermined transmitted data, which is the first and second branch metric values bmcl and bmc2.

[0040] The previous metric value PM1 is read from the metric memory 3b. The number and length of the states may be changed according to use. Also, the size of the first and second registers 10A and 10B and the first and second adders 30 and 35 are determined accordingly.

[0041] As expressed in the above formula, in the case of a state 00, the previous metric value $M_{n-1}{}^{p0}$ is read. Then, the value obtained by adding the read value with $bmc1^{p0}{}_{s0}$ are compared with the value obtained by adding metric value $M_{n-1}{}^{p1}$, in effect the probability of being in a state p1(01) in the (n-1)th cycle having followed a given path, and $bmc2^{p1}{}_{s0}$, the probability of a transition to state s0(00) from state p1(01). Of the two values, the larger value is determined as the survival metric M_n,s0 of the state 00.

[0042] The information indicating which of the two survives, i.e., a determining bit, is generated by the comparator 40. The determining bit (SEL) is stored in the path memory 4b and is to be used later in tracing back data.

[0043] The gist of the present invention is in that the data, and in particular previous metric values, such as $M_{n-1}{}^{p0}$, is read from a previous metric memory and stored in the register portion 10 to be used in calculating the next state, without the need to use the same memory repeatedly. Referring to FIG. 2, the metric values of the previous states 00 and 01 are read in both of the present states 00 and 10, respectively. Therefore, the sequence of calculating the present metric states is from 00 to 10 to 01 to 11, unlike the conventional sequence from 00 to 01 to 10 to 11. In other words, the sequence of reading the previous metric memory is from the state 00 to 01 (for present state 00) to 00 to 01 (for present state 10) to 10 to 11 (for present state 01) to 10 to 11 (for present state 11). Thus, the metric values corresponding to the previous states 00 and 01 read from the metric memory 3b are latched to the first and second registers 10B and 10A of the register portion 10 respectively and then the ACS values of the present states 00 and 10 are sequentially calculated. Thus, after calculating the present state 00, the metric values corresponding to the previous states 00 and 01 are read from the first and second registers 10B and 10A, thus calculating the ACS output value of the present state 10, without the need to read the metric memory 3b again.

[0044] The metric values of the previous states 10 and 11 are latched from the metric memory 3b and then stored in the first and second registers 10B and 10A. Then, the present states 01 and 11 are sequentially calculated in a similar manner to that described above. Thus, the number of times the previous metric values are read from the metric memory 3b is reduced from 8 times to 4 times.

[0045] FIG. 5 is an operational timing diagram according to a preferred embodiment of the present invention, in which FIG. 5A illustrates the sequence of reading previous metric values from the metric memory 3b, FIG. 5B illustrates the value of the first register 10B in the register portion 10, FIG. 5C illustrates the value of the second register 10A, FIG. 5D illustrates the value of bmcl, FIG. 5E illustrates the value of bmc2 and FIG. 5F illustrates the present metric value stored in the path memory 4b.

[0046] As shown by periods T1 and T2 in FIG. 5A, the metric values $M_{n-1}{}^{p0}$ and $M_{n-1}{}^{p1}$ for the previous states 00 and 01 are sequentially read from the metric memory 3b. Then, as shown by block T1 in FIG. 5B and block T2 in FIG. 5C, the values read are stored in the first and second registers 10A and 10B respectively. As shown by block T1 in FIGs. 5D and 5E, using the values $bmc1^{p0}{}_{s0}$ and $bmc2^{p1}{}_{s0}$, calculated by the BMC 1b, the ACS values are calculated based on the above formula (1). The resulting present metric value of the present state 00 is M_n,s0, as shown by blocks T1 and T2 in FIG. 5F.

[0047] Next, the metric values $M_{n-1}{}^{p0}$ and $M_{n-1}{}^{p1}$ for the previous states 00 and 01, stored in the first and second registers 10B and 10A, and the values $bmc1^{p0}{}_{s2}$ and $bmc2^{p1}{}_{s2}$, calculated by the BMC 1b and shown by block T3 in FIGs. 5D and 5E, are used to calculate the ACS values based on the above formula (1). The resulting present metric value of the present state 10 is M_n,s2, as shown by block T3 in FIG. 5F.

[0048] As shown by blocks T4 and T5 in FIG. 4A, the metric values $M_{n-1}{}^{p2}$ and $M_{n-1}{}^{p3}$ for the previous states 10 and 11 are sequentially read from the metric memory 3b. Then, as shown by block T4 in FIG. 5B and block T5 in FIG. 5C, the values read are stored in the first and second registers 10A and 10B respectively. As shown by block T4 in FIGs. 5D and 5E, using the values $bmc1^{p0}{}_{s0}$ and $bmc2^{p1}{}_{s0}$, calculated by the BMC 1b, the ACS values are calculated based on the above formula (1). The resulting present metric value of the present state 01 is M_n,s1, as shown by blocks T4 and T5 in FIG. 5F.

[0049] Next, the metric values $M_{n-1}{}^{p2}$ and $M_{n-1}{}^{p3}$ for the previous states 10 and 11, stored in the first and second registers 10B and 10A, and the values $bmc1^{p2}{}_{s3}$ and $bmc2^{p3}{}_{s3}$, calculated by the BMC 1b and shown by block T6 in FIGs. 5D and 5E, the ACS values are calculated based on the above formula (1). The resulting present metric value of the present state 11 is M_n,s3, as shown by block T6 in FIG. 5F.

[0050] In conclusion, as described above, the sequence of calculating the present metric values is changed, i.e., 00 to 10 to 01 to 11. Thus, when obtaining the metric values of the present states 10 and 11, it is not necessary to access

the memory again to read the previous metric values. Instead, the previous metric values read for obtaining the metric values of the present states 00 and 01 are used, thus saving time.

[0051] In other words, in the case of four states, the number of times the metric memory 3b is read is conventionally 8 times, but 4 times according to the present invention. Also, the number of overall required clocks is reduced by 2 clock cycles. Thus, 8 clock cycles are required conventionally, but only 6 clock cycles are required according to the present invention. The blocks T1 through T6 correspond to unit clock cycles.

[0052] FIG. 6 illustrates an exemplary Viterbi equalizer according to an embodiment of the present invention. The filter 10 and the impulse response estimater 20 represented by a block 100 is a preprocessing portion and is implemented by the DSP. A portion represented by reference numeral 200 is a pure Viterbi algorithm processing portion and is implemented by hardware.

[0053] In FIG. 6, the Viterbi algorithm processing apparatus 25 has the same structure as that shown in FIG. 3B. An EDC (Euclidian Distance Calculation) Ia in FIG. 3A implements the known calculations given by

$$edc^{p0}_{sk(k=0,2)} = \left|r_n - r^{p0}_{sk}\right|^2 = \left[R(r_n) - R(r^{p0}_{sk})\right]^2 + \left[I(r_n) - I(r^{p0}_{sk})\right]$$

$$edc^{p2}_{sk(k=1,3)} = \left|r_n - r^{p2}_{sk}\right|^2 = \left[R(r_n) - R(r^{p2}_{sk})\right]^2 + \left[I(r_n) - I(r^{p2}_{sk})\right]$$

$$edc^{p1}_{sk(k=0,2)} = \left|r_n - r^{p1}_{sk}\right|^2 = \left[R(r_n) - R(r^{p1}_{sk})\right]^2 + \left[I(r_n) - I(r^{p1}_{sk})\right]$$

$$edc^{p3}_{sk(k=1,3)} = \left|r_n - r^{p3}_{sk}\right|^2 = \left[R(r_n) - R(r^{p3}_{sk})\right]^2 + \left[I(r_n) - I(r^{p3}_{sk})\right]$$

where R is a real number, I is an imaginary number, r_n is received data, and $r_n{}^{p1}$ and $r_n{}^{p2}$ are reference data.

[0054] To perform the Euclidian calculation, the EDC 1a requires subtracters, multipliers and adders. After Euclidian calculation, the ACS 2a performs the known add-compares calculation given by

$$M_n,s0 = \min(M_{n-1}{}^{p0} + edc1^{p0}{}_{s0},\ M_{n-1}{}^{p1} + edc2^{p1}{}_{s0})$$

$$M_n,s1 = \min(M_{n-1}{}^{p2} + edc1^{p2}{}_{s1},\ M_{n-1}{}^{p3} + edc2^{p1}{}_{s1})$$

$$M_n,s2 = \min(M_{n-1}{}^{p0} + edc1^{p0}{}_{s2},\ M_{n-1}{}^{p1} + edc2^{p1}{}_{s2})$$

$$M_n,s3 = \min(M_{n-1}{}^{p2} + edc1^{p2}{}_{s3},\ M_{n-1}{}^{p3} + edc2^{p1}{}_{s3})$$

[0055] The order in which these calculations can be carried out can be the same as described in connection with branch metric calculators. Indeed by replacing bmcl and bmc2 with edc1 and edc2, the apparatus of fig. 4 can be used for this purpose.

[0056] For the add-compare calculation, the ACS 2a requires adders and comparators. Like the Euclidian calculation, the add-compare calculation also requires calculations (additions) twice in the respective states ie to calculate the M_ns0 and M_ns2 or M_ns1 and M_ns3. Therefore, it is possible to increase the data processing speed by implementing this portion ie ACS 2a by hardware to operate in parallel. In other words, the conventional apparatus implemented by the DSP performs the calculations in sequence, which takes more time, as compared with the case where the calculations are performed in parallel. In the embodiment, the portion for processing the pure Viterbi algorithm is implemented by hardware, so that about 1/5 cycle time is required as compared with the case where it is implemented by the DSP, thereby reducing the power consumption and securing the time margin in implementing the system.

[0057] As described above, according to the present invention, overall Viterbi processing time can be reduced (e.g.

by 25% when using an ACS block as in the described embodiment of the present invention). In other words, instead of the conventional ACS calculating method, by changing the sequence of reading the memory, the overall number of memory accesses and memory access time can be reduced, which allows for a certain timing margin in implementing the system, thus increasing reliability. Also, power consumption can be reduced by reducing the number of times the memory is read.

[0058] Furthermore, by divisionally implementing the Viterbi equalizer and the Viterbi decoder into a portion to be processed by the DSP and a portion to be processed by the hardware, it is possible to consider the particulars pertinent to the size of the mobile communication terminal at the design state. In addition, the portion implemented by hardware has a parallel data processing path, increasing the data processing speed.

Claims

1. A high-speed add-compare selection apparatus, for a Viterbi algorithm processing apparatus having a branch metric calculator or a Euclidean value calculator or the like and a metric memory, comprising:

   first and second registers;
   means for supplying to the first and second registers first and second previous metric values from the metric memory;
   a first adder for adding the first previous metric value from the first register and a branch metric value or a Euclidean value or the like of the present state calculated by the calculator;
   a second adder for adding the second previous metric value from the second register and a branch metric value or a Euclidean value or the like of the next state calculated by the calculator; and
   a comparator for comparing the outputs of the first and second adders and calculating a survival metric value accordingly.

2. A high-speed add-compare selection apparatus according to claim 1 in which the means for supplying the first and second previous metric values to the first and second registers comprises means for supplying to the first register a first previous metric value from the second register and for supplying to the second register a second previous metric value from the metric memory.

3. A high-speed add-compare selection apparatus according to claim 1 or claim 2 in which the first and second metric values are Hamming distances.

4. A high-speed add-compare selection apparatus according to claim 1 or claim 2 in which the first and second metric values are Euclidian distances.

5. A high-speed add-compare selection apparatus according to any preceding claim in which the first adder is for adding the first previous metric value input from the first register to a Euclidean value of a next state calculated by the Euclidean value calculator and the second adder is for adding the second previous metric value input from the second register to the Euclidean value of the next state calculated by the Euclidean value calculator.

6. A Viterbi algorithm processing method for use in a processing apparatus having a branch metric calculator or a Euclidean value calculator or the like and a metric memory, comprising:

   supplying to first and second registers first and second previous metric values from the metric memory;
   calculating a first survival metric value of a present state using the first and second previous metric values from the first and second registers, a branch metric value or Euclidean value or the like of the present state calculated by the calculator and a branch metric value or Euclidean value or the like of the next state calculated by the calculator;
   calculating a second survival metric value of another present state using the first and second previous metric values from the first and second registers, a branch metric value or a Euclidean value or the like of the present state calculated by the calculator and a branch metric value or a Euclidean value or the like of the next state calculated by the calculator;
   supplying to the first and second registers third and fourth previous metric values from the metric memory; and
   calculating a third survival metric value of another present state using the third and fourth previous metric values from the first and second registers, a branch metric value or Euclidean value or the like of the present state calculated by the calculator and a branch metric value or a Euclidean value or the like of the next state

calculated by the calculator.

7. A Viterbi algorithm processing method according to claim 6 further comprising:
calculating a fourth survival metric value of another present state using the third and fourth previous metric values from the first and second registers, a branch metric value or Euclidean value or the like of the present state calculated by the branch metric calculator and a branch metric value or Euclidean value or the like of the next state calculated by the calculator.

8. A Viterbi algorithm processing method according to claim 6 or claim 7 in which the survival metric values of the present states are calculated by:

adding the previous metric value from the first register and a branch metric value or Euclidean value or the like of the present state calculated by the calculator to provide a first added value;
adding the previous metric value from the second register and a branch metric value or a Euclidean value or the like of the next state calculated by the calculator to provide a second added value; and
comparing the first and second added values and calculating a survival metric value accordingly.

9. A Viterbi algorithm processing method for use in a processing apparatus having a branch metric or Euclidean value calculator or the like, a metric memory, first and second registers and an add-compare selector for comparing a value obtained by adding a first previous metric value to a branch metric or Euclidean value or the like of a present state calculated by the calculator with a value obtained by adding a second metric value to a branch metric, Euclidean value or the like of a next state calculated by the calculator to determine a higher value of them as a survival metric, comprising the steps of:

(a) reading the first and second previous metric values from the metric memory and storing the read values in the first and second registers to calculate a metric value of the present state;

(b) calculating a metric value of another present state using the first and second previous metric values of the first and second registers; and

(c) returning to said step (a) to calculate a metric value of another present state using a previous metric value being different from the first and second metric values of the first and second registers.

10. A Viterbi algorithm processing method for use in an add-compare selector having first and second registers, a branch metric or Euclidean value calculator or the like and a metric memory, comprising the steps of:

(a) reading n-th (where n = 1, 2, 3,....) and (n+1)-th previous metric values from the metric memory and storing the read values in the first and second registers;

(b) comparing a value obtained by adding the n-th previous metric value stored in the first register to a branch metric valve, Euclidean value or the like of the present state calculated by the calculator with a value obtained by adding the (n+1)-th previous metric value stored in the second register to a branch metric, Euclidean value or the like of a next state calculated by the calculator so as to determine a higher value of them as an n-th survival metric;

(c) comparing a value obtained by adding the n-th previous metric value stored in the second register to another branch metric or Euclidean value or the like of the present state calculated by the calculator with a value obtained by adding the (n+1)-th previous metric value stored in the second register to another branch metric valve or Euclidean value or the like of the next state calculated by the calculator so as to determine a higher value of them as an (n+1)-th survival metric; and

(d) increasing n by one and returning to said step (a) to process the Viterbi algorithm for another present state.

11. A Viterbi equalizer comprising:

a Viterbi pre-processing portion for estimating a channel impulse response from received data and for filtering the data using the impulse response using a digital signal processor; and

a Viterbi algorithm processing apparatus for performing a Viterbi algorithm for equalization of the filtered data and the estimated channel impulse response provided from the Viterbi preprocessing portion.

12. An equalizer according to claim 11 in which the pre-processing portion multiplies a reversal of the estimated channel impedance response by the received data, and time-shifts the multiplied result.

13. A Viterbi equalizer according to claim 11 or 12 comprising a high speed add-compare selection apparatus according to any of claims 1 to 5 or such an apparatus adapted to operate the method of any of claims 6-8.

14. A high-speed add-compare selection apparatus, for a Viterbi algorithm processing apparatus having a branch metric calculator and a metric memory, as described with reference to and as illustrated in the accompanying drawings.

15. A Viterbi equalizer as described with reference to and as illustrated in the accompanying drawings.

16. A Viterbi algorithm processing method as described with reference to and as illustrated in the accompanying drawings.

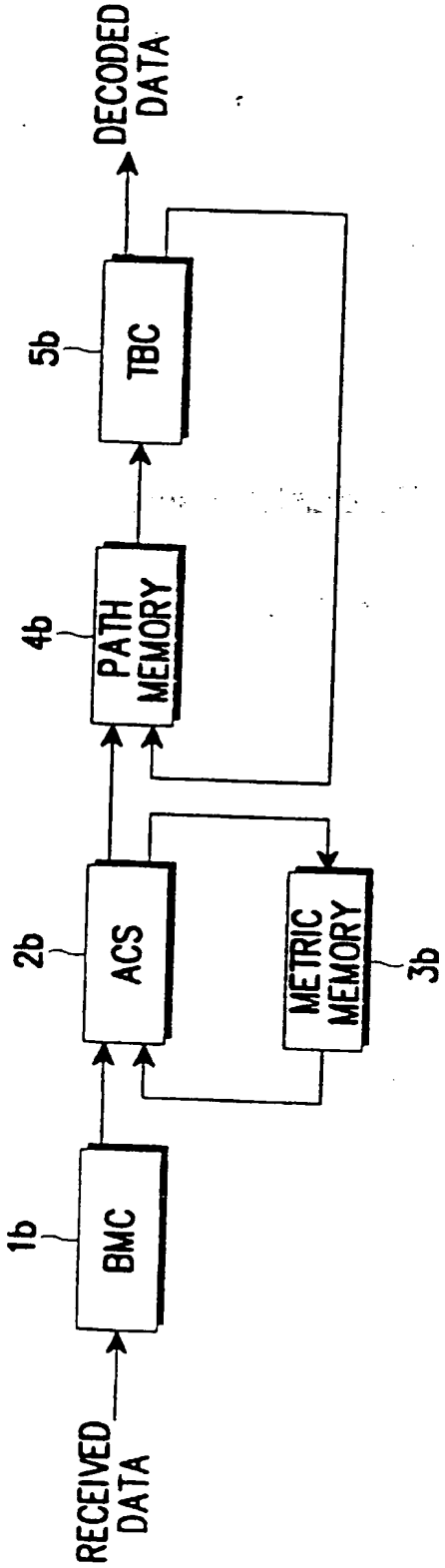
RECEIVED DATA
1b
BMC
2b
ACS
3b
METRIC MEMORY
4b
PATH MEMORY
5b
TBC
DECODED DATA

FIG. 1

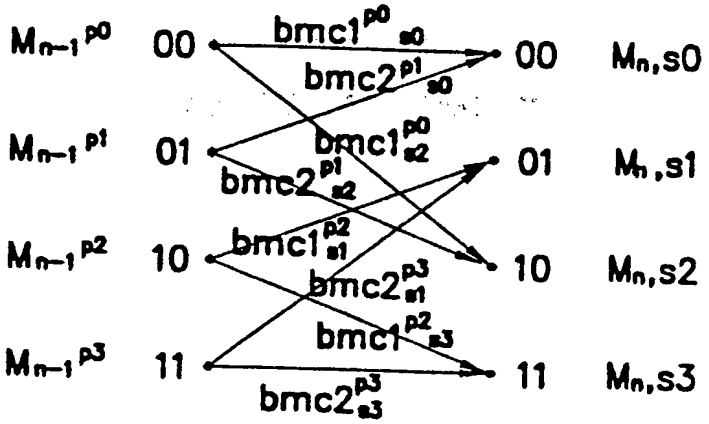
$M_{n-1}{}^{p0}$
00
$M_{n-1}{}^{p1}$
01
$M_{n-1}{}^{p2}$
10
$M_{n-1}{}^{p3}$
11
$bmc1^{p0}_{s0}$
$bmc2^{p1}_{s0}$
$bmc1^{p0}_{s2}$
$bmc2^{p1}_{s2}$
$bmc1^{p2}_{s1}$
$bmc2^{p3}_{s1}$
$bmc1^{p2}_{s3}$
$bmc2^{p3}_{s3}$
00
01
10
11
M_n,s0
M_n,s1
M_n,s2
M_n,s3

FIG. 2

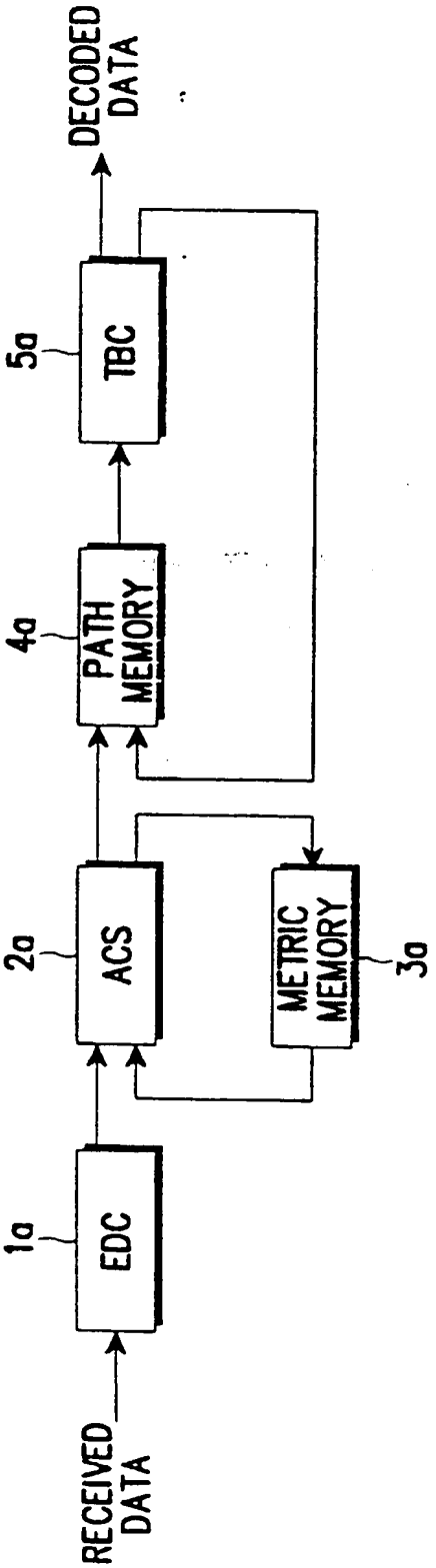
DECODED DATA
5a
TBC
4a
PATH MEMORY
2a
ACS
METRIC MEMORY
3a
1a
EDC
RECEIVED DATA

FIG. 3A

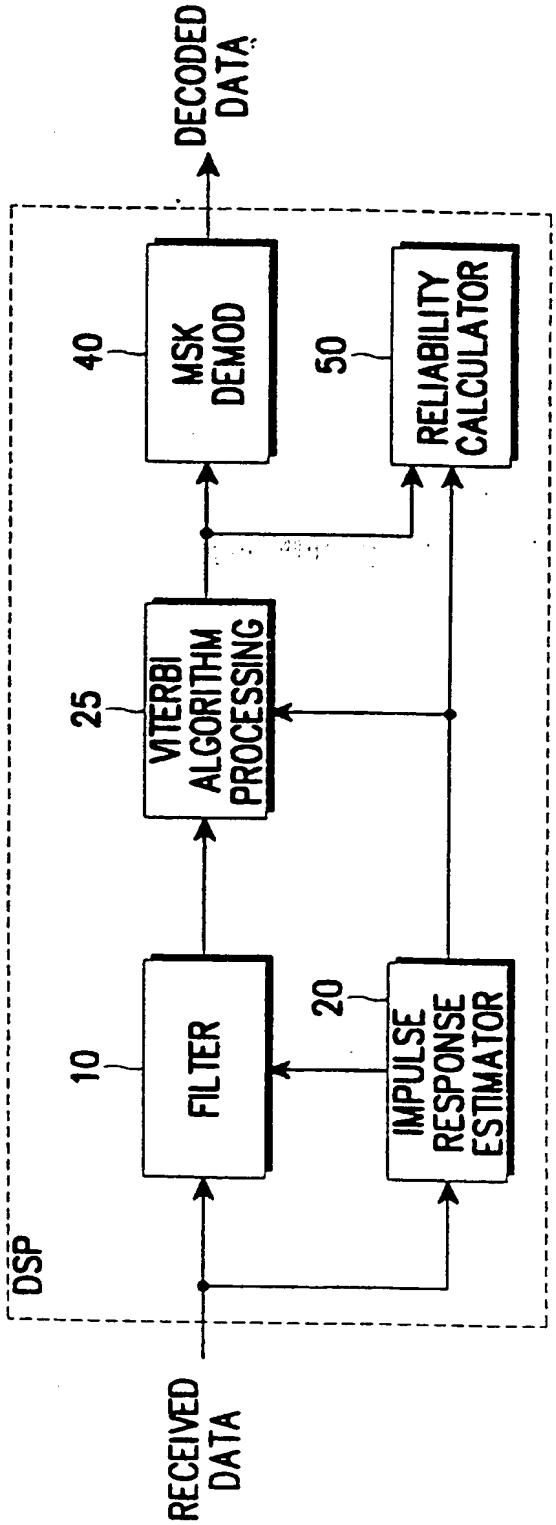
DECODED DATA
40
MSK DEMOD
50
RELIABILITY CALCULATOR
25
VITERBI ALGORITHM PROCESSING
10
FILTER
20
IMPULSE RESPONSE ESTIMATOR
DSP
RECEIVED DATA

FIG. 3B

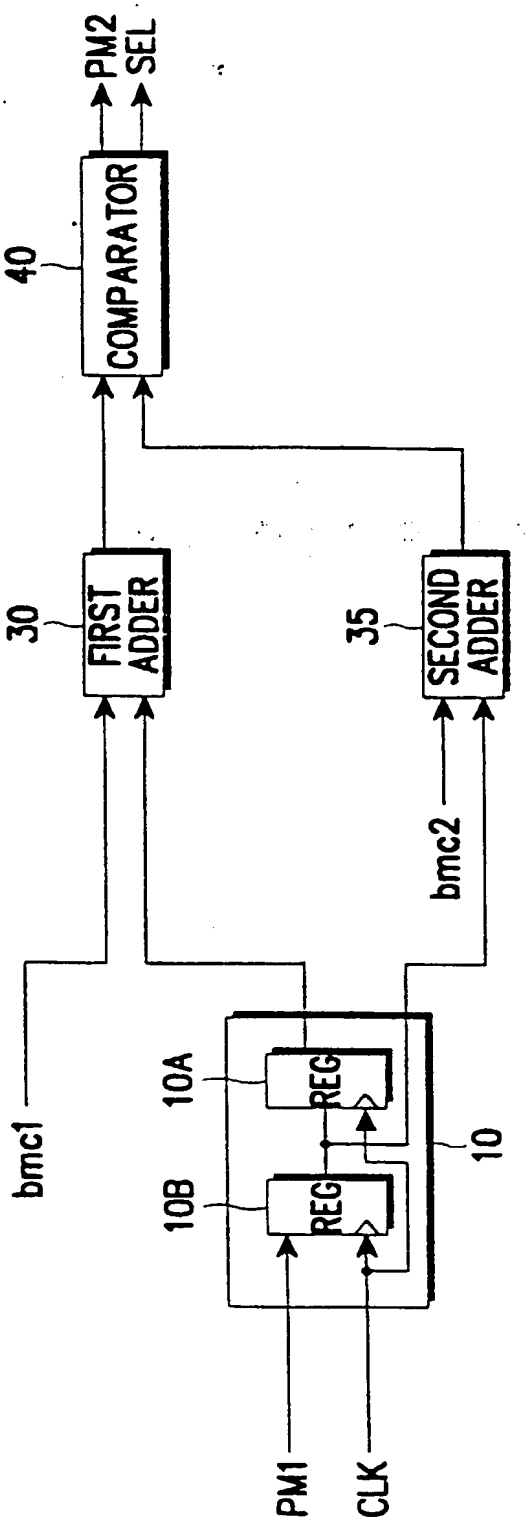
PM2
SEL
40
COMPARATOR
30
FIRST ADDER
35
SECOND ADDER
bmc2
bmc1
10A
REG
10B
REG
10
PM1
CLK

FIG. 4

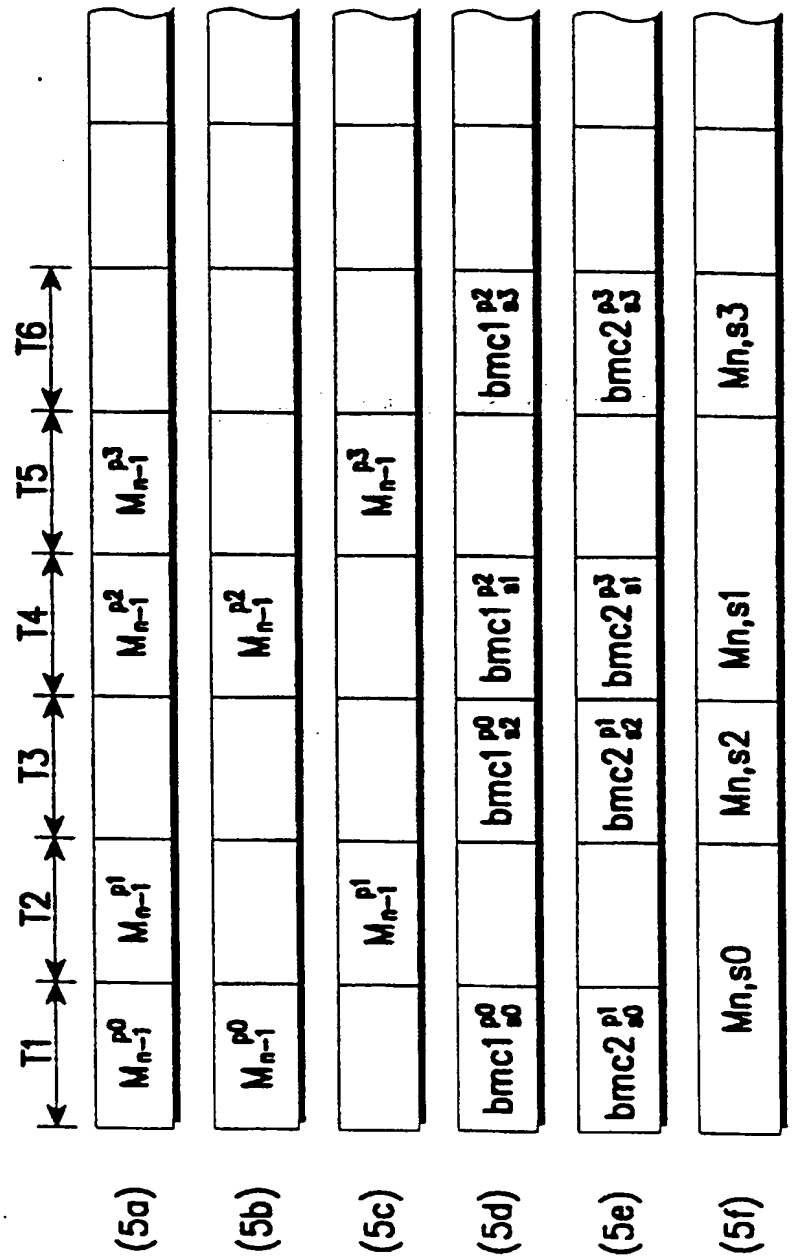

FIG. 5

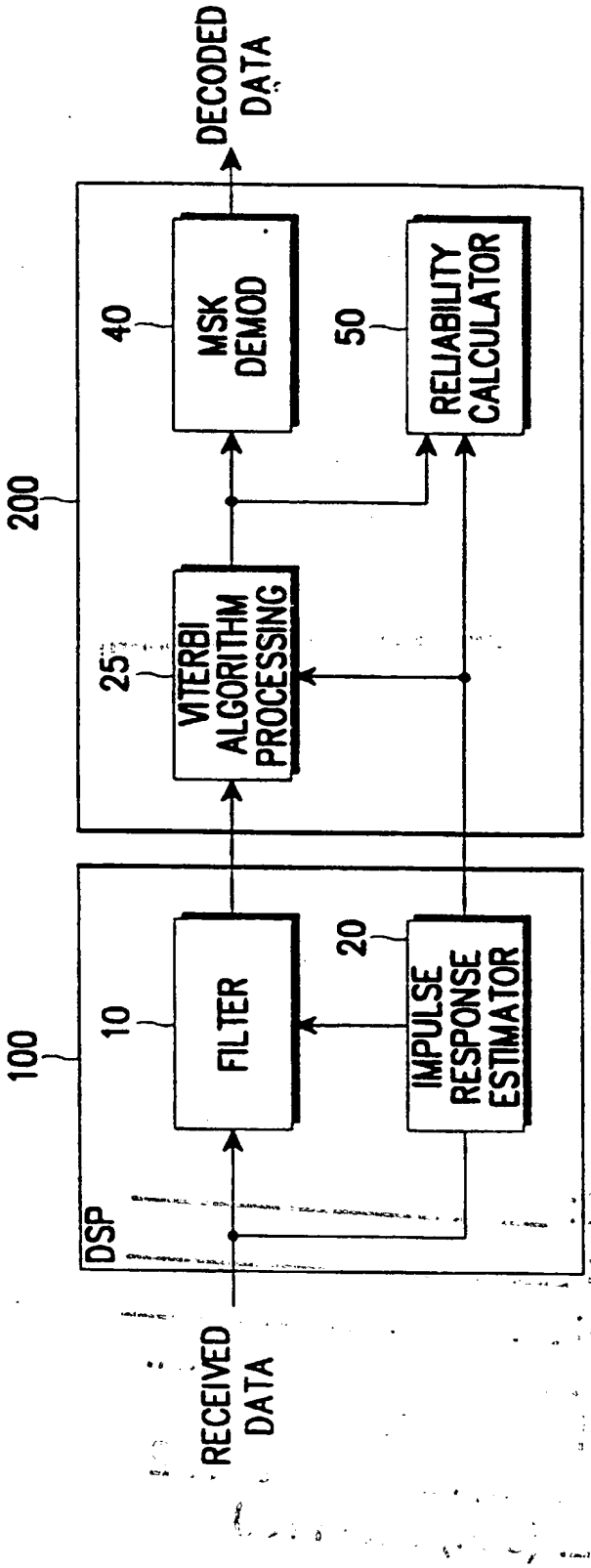
DECODED DATA
MSK DEMOD
40
RELIABILITY CALCULATOR
50
200
VITERBI ALGORITHM PROCESSING
25
FILTER
10
IMPULSE RESPONSE ESTIMATOR
20
100
DSP
RECEIVED DATA

FIG. 6

THIS PAGE BLANK (USPTO)

DOCKET NO: GR 00 P 1969
SERIAL NO: 09/864,980
APPLICANT: Aymar, et al.

LERNER AND GREENBERG P.A.
P.O. BOX 2480
HOLLYWOOD, FLORIDA 33022
TEL. (954) 925-1100